Q1. What are the benefits of the built-in array package, if any?

Answer:

Arrays are a type of data structure in which multiple elements of the same type are represented by a single name. The elements are accessed randomly using the index number, and the arrays allocate memory in contiguous locations for all elements. This eliminates the possibility of additional memory being allocated in the case of arrays, thus avoiding memory overflow or shortage in arrays. As a result, lists and numPy Arrays are often preferred. This is one of the reasons why few people are aware that Python has an implicit Array type. However, when something is limited, it must provide advantages. This is known as a "trade-off". In this example, the size of the Python Array is smaller than that of the Python List.

|  |
| --- |
| from array import array  list\_ex = list(range(0, 100000))  arr\_ex = array('I', list\_ex)  list\_ex.\_\_sizeof\_\_() # Size comes out to be 800040  arr\_ex.\_\_sizeof\_\_() # Size comes out to be 400064 |

Q2. What are some of the array package's limitations?

Answer:

* It is necessary to know in advance the number of elements to store in an array.
* An array is an immutable structure (i.e., the size cannot be changed once declared).
* The memory that is allocated to the array cannot be incremented or decreased.
* The insertion and deletion of elements in an array is problematic as they are stored in successive memory locations and the operation of shifting is expensive.
* If the memory is allocated more than the requirement, it will result in a waste of memory space, and if the memory is allocated less than the requirement, a problem will arise.

Q3. Describe the main differences between the array and numpy packages.

Answer:

* The array package does not help with numerical calculation by insiding it in number form, whereas NumPy provides with a wide range of numerical operations.
* An array is a single-dimensional entity that stores the numerical data, whereas numpy can handle more than one dimension.
* In the case of an array, an item can be accessed through its index position, which is easy to do, whereas in numpy, an item is accessed via its column and row index, which is slightly time consuming. The same is true for appending operation.
* In an array, we do not create a tabular structure while in numpy, it creates a tabular structure.

Q4. Explain the distinctions between the empty, ones, and zeros functions.

Answer:

Empty -> A function definition that does not include any statement in its body is referred to as an empty function. Writing a function definition in python without any statement will result in an error. In order to avoid this, a pass statement is used. A pass statement in Python is a dummy statement that does not do anything.

Ones -> This function returns the value of the element in a new array of the shape and data type. The value of the element is 1.

Zeros -> The function returns a new set of the shape and the data type, the value of which is 0.

Q5. In the fromfunction function, which is used to construct new arrays, what is the role of the callable argument?

Answer:

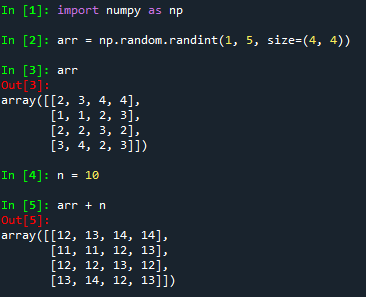
The primary purpose of this function is to perform the function on each coordinate and on the resultant array. This function is invoked with the parameter N, where N represents the rank of the shape. The parameter represents the coordinates of the array varying along a particular axis.

Q6. What happens when a numpy array is combined with a single-value operand (a scalar, such as an int or a floating-point value) through addition, as in the expression A + n?

Answer:

If we add a scalar value like integer to a numpy array, all the elements in the array will also add that value.

Example:



Q7. Can array-to-scalar operations use combined operation-assign operators (such as += or \*=)? What is the outcome?

Answer:

It performs the operation according to operators. It performs the specified operation on all the elements of the array. For example, if + operand is used, the current array will be updated by addition, and if '\*' is used, it will be updated by multiplication.

Q8. Does a numpy array contain fixed-length strings? What happens if you allocate a longer string to one of these arrays?

Answer:

It is possible to include a fixed length string in numpy arrays. The length of the dtype of a numpy array that contains string values is the length of any string in the array. Once set, it will be able to store only new strings with a length that is not greater than the length of the string at the time of its creation. If we attempt to reassign another string value with a length greater than the existing elements' length, it will simply discard all the values that exceed the maximum length accepted up to those values that are below the limit.

Q9. What happens when you combine two numpy arrays using an operation like addition (+) or multiplication (\*)? What are the conditions for combining two numpy arrays?

Answer:

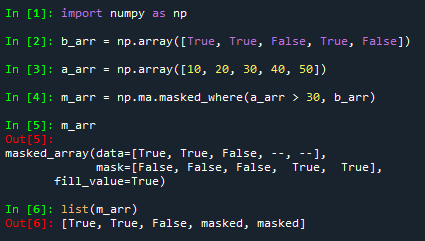
It simply multiplies or adds an element to an element at the same position. The only conditions that need to be satisfied are: (a) the data type must be the same; and (b) the shape of both matrices must match.

Q10. What is the best way to use a Boolean array to mask another array?

Answer:

The best method of masking another array by using a Boolean array is through the use of the numpy package's masked\_where function.

Example:



Q11. What are three different ways to get the standard deviation of a wide collection of data using both standard Python and its packages? Sort the three of them by how quickly they execute.

Answer:

Following are the different ways to get the standard deviation of a wide collection of data using both standard Python and its packages:

* std() method from numpy package
* stdev() method from statistics package
* custom method can be developed to calculate standard deviation using standard Python

Example

|  |
| --- |
| def calc\_average(arr):  n = len(arr)  avg = sum(arr)/n  return avg  def calc\_variance(arr):  n = len(arr)  avg = calc\_average(arr)  deviation = [(i - avg)\*\*2 for i in arr]  variance = sum(deviation)/n  return variance  def calc\_std(arr):  variance = calc\_variance(arr)  std = variance\*\*(1/2)  return std  arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])  import numpy as np  import statistics as st  print('numpy-std: ', np.std(arr))  print('statistics-stdev', st.stdev(arr))  print('custom-std', calc\_std(arr))  ## output  # numpy-std: 2.8722813232690143  # statistics-stdev 3.0  # custom-std 2.8722813232690143 |

12. What is the dimensionality of a Boolean mask-generated array?

Answer:

It will have the same dimensions as the input array. Masking occurs when a value is to be extracted, changed, counted, or otherwise manipulated in an array according to a certain criteria: for instance, counting all values that exceed a certain threshold value, or removing all outliers above a certain threshold value. In the case of Boolean masking in NumPy, the most efficient approach is often to use Boolean masking.